VERIFICATION ISSUES FOR RULE-BASED EXPERT SYSTEMS

Chris Culbert, Gary Riley, Robert T. Savely
Artificial Intelligence Section - FM72
NASA/Johnson Space Center
Houston, TX 77058

ABSTRACT

Expert systems are a highly useful spinoff of the artificial intelligence research efforts. One major stumbling block to extended use of expert systems is the lack of well-defined verification and validation (V&V) methodologies. Since expert systems are computer programs, the definitions of "verification" and "validation" from conventional software are applicable. The primary difficulty with expert systems is the use of development methodologies which don't support effective V&V. If proper techniques are used to document requirements, V&V of rule-based expert systems is possible, and may be easier than with conventional code. For NASA applications, the flight technique panels used in previous programs should provide an excellent way of verifying the rules used in expert systems. There are, however, some inherent differences in expert systems that will affect V&V considerations.

INTRODUCTION

Expert systems represent one important by-product of Artificial Intelligence research efforts. They have been under development for many years and have reached commercial viability in the last three to four years. However, despite their apparent utility and the growing number of applications being developed, not all expert systems reach the point of operational use. One reason for this is the lack of well understood techniques for V&V of expert systems.

Developers of computer software for use in mission or safety critical applications have always relied upon extensive V&V to ensure that safety and/or mission goals were not compromised by software problems. Expert system applications are computer programs and the same definitions for V&V apply to expert systems. Consequently, expert systems require the same assurance of correctness as conventional software.

Despite the clear need for V&V, considerable confusion exists over how to accomplish V&V of an expert system. There are even those who question whether or not it can be done. This confusion must be resolved if expert systems are to succeed. As with conventional software, the key to effective V&V is through the proper use of a development methodology which both supports and encourages the development of verifiable software.

THE COMMON EXPERT SYSTEM DEVELOPMENT METHODOLOGY

Most existing expert systems are based upon relatively new software techniques which were developed to describe human heuristics and to provide a better model of complex systems. In expert system terminology, these techniques are called knowledge representation. Although numerous knowledge representation techniques

are currently in use (rules, objects, frames, etc) they all share some common characteristics. One shared characteristic is the ability to provide a very high level of abstraction. Another is the explicit separation of the knowledge which describes how to solve problems from the data which describes the current state of the world.

Each of the available representations have strengths and weaknesses. With the current state-of-the-art, it is not always obvious which representation is most appropriate for solving a problem. Therefore, most expert system development is commonly done by rapid prototyping. The primary purpose of the initial prototype is to demonstrate the feasibility of a particular knowledge representation. It is not unusual for entire prototypes to be discarded if the representation doesn't provide the proper reasoning flexibility.

Another common characteristic of expert system development is that relatively few requirements are initially specified. Typically, a rather vague, very general requirement is suggested, e.g., "We want a program to do just what Charlie does". Development of the expert system starts with an interview during which the knowledge engineer tries to discover both what it is that Charlie does and how he does it. Often there are no requirements written down except the initial goal of "doing what Charlie does". All the remaining system requirements are formulated by the knowledge engineer during development. Sometimes, the eventual users of the system are neither consulted nor even specified until late in the development phase. As with conventional code, failure to consult the intended users early in the development phase results in significant additional costs later in the program.

So where does all this lead? The knowledge engineer is developing one or more prototypes which attempt to demonstrate the knowledge engineer's understanding of Charlie's expertise. However, solid requirements written down in a clear, understandable, *easy to test* manner generally don't exist. This is why most expert systems are difficult to verify and validate; not because they are implicitly different from other computer applications, but because they are commonly developed in a manner which makes them very difficult or impossible to test.

NEW APPROACHES TO DEVELOPMENT METHODOLOGIES

From the preceding section, it should be clear that the problem is the use of development methodologies which generally do not generate requirements which can be tested. Therefore, the obvious solution is to use a methodology which will produce written requirements which can be referred to throughout development to verify correctness of approach and which can be tested at the end of development to validate the final program.

Unfortunately, it's not that simple. Some expert systems can probably be developed by using conventional software engineering techniques to create software requirements and design specifications at the beginning of the design phase [1]. However, the type of knowledge used in other expert systems doesn't lend itself to this approach. It is best obtained through iterative refinement of a prototype which allows the expert to spot errors in the expert system reasoning before he can clearly specify the correct rules.

The goal of any software development methodology is to produce reliable code that is both maintainable and verifiable. A software development methodology for expert systems must serve a similar purpose as one for conventional software. However, there are some differences between expert systems and conventional software which will affect the development methodology. Development methodologies for expert systems are discussed in more detail in another paper by the authors [2]. Suffice to say here that some kind of development methodology must be chosen and applied to support effective V&V.

MAKING THE REQUIREMENTS WORK

Once we accept that requirements and specifications must be written and a methodology for how and when to write them has been adopted, the actual work of verifying and validating the program must be done. A very appropriate technique would be a direct derivative of the methods used to develop procedures, flight rules, and flight software for the Apollo and Shuttle programs. This technique consists of Flight Technique Panels which regularly review both the procedures for resolving a problem and the analysis techniques used to develop those procedures.

If expertise is not readily available from past experience, the analysis efforts typically use high fidelity simulations based on system models to derive and evaluate control parameters. If expertise is available, the knowledge is reviewed by the panel and placed in the appropriate context. The panels consist of system users, independent domain experts, system developers, and managers to ensure adequate coverage of all areas of concern. In previous programs, the typical output of such a panel was a set of flight rules describing the operational requirements for a system.

Sometimes these flight rules were translated into computer programs (typically as decision trees) and embedded in the onboard or ground computers. An additional verification step was needed to guarantee that the flight rules approved by the panel were properly coded. More often, computer limitations caused the flight rules to remain in document form used directly by flight controllers and mission crews.

For future programs, many of the flight rules which come from the Flight Technique Panels can be coded directly into expert systems. Expert systems developed in this manner will have undergone extensive verification through the panel review. They should also prove easier to verify in code form because the rule language will allow the program to closely resemble the original flight rule.

Programs of the complexity and size with which NASA regularly deals make this approach mandatory. Smaller programs generally will not require the resources or effort involved in verifying a system to this extent. The size of the panel and the length of the review process can be scaled down to something appropriate for the complexity and size of the application. For some applications, the panel approach could look very similar to independent code review techniques.

Exhaustive testing through simulation remains the most effective method available for final validation. However, for any system of reasonable complexity, exhaustive testing is both prohibitively expensive and time consuming. Space Shuttle applications typically used extensive testing with data sets representative of the

anticipated problems or failure modes. This method is not guaranteed to eliminate all software bugs, but it can prevent the *anticipated* problems. If used properly, representative testing can eliminate enough problems to make the software acceptable for mission and safety critical applications.

The panel approach to verification discussed above is very effective at ensuring that the knowledge in the expert system is both correct and complete. Verification of conventional software also covers feasibility, maintainability, and testability. These verification efforts are generally done early in the design phase and should also be done for an expert system. The coded rules must also be examined to ensure that the consistency and completeness of the design is properly incorporated in the software.

Some of this work can be done automatically. Testing a rule language for completeness and consistency may actually be easier than testing conventional programs. The explicit separation of knowledge elements from control and data elements may allow relatively straightforward analysis of the rules by automated tools [3]. If automated methods are not used, other standard methods such as code reviews and manual examination of the rules may also be comparatively easy, again due to the independent nature of the knowledge elements. They can be done by the whole panel, or more likely, small teams of people drawn from the whole panel.

Feasibility of knowledge representation is usually fully tested in the early prototypes, but the feasibility of other elements of the expert system, such as performance, user interfaces, data interfaces, etc. must also be verified. The use of rapid prototyping can be extended from testing representation to testing some of these areas as well. Iterative development can go a long way to ensuring that the final system truly meets the user needs in these kind of areas.

Finally, the requirements must be examined to ensure that they are able to be tested. They should be specific, unambiguous and quantitative where possible. Objective requirements will aid in the development of rigorous test cases for final validation. A test plan should be written which discusses how the final expert system will be tested.

OTHER ISSUES FOR EXPERT SYSTEM V&V

There are other differences between between conventional software and expert systems, and those differences will affect V&V efforts. Some of the differences are discussed in reference [4] and summarized below.

Verifying the Correctness of Reasoning

Verifying that an expert system solves a problem for the right reasons is sometimes as important as getting the right answer. For a rule-based expert system, identifying all possible paths to a solution is very difficult. Therefore, it is important to ensure that the expert system has gotten the right answer for the right reasons.

Verifying the Inference Engine

The inference engine in a rule-based expert systems is a completely separate piece of code anc can be fully verified independently from the rest of the expert system.

Verifying the Expert

This question is automatically resolved as long as the expert system is validated. The panel approach discussed in this paper provides continual feedback on the correctness of the experts knowledge.

Real-Time Performance

Most conventional programs provide performance "guarantees" through extensive simulation of the expected performance environment. Expert systems can provide the same kind of performance "guarantees". Some kinds of conventional programs are analyzed at the machine instruction level to specifically determine the amount of time required to process a given data set. Achieving the same kind of capability in a rule-based expert system is more difficult, but can be done for a given data set entered in a specific sequence.

Complex Problems with Multiple Experts

The panel review method already discussed here is clearly the appropriate method for resolving a problem of this type. The review process used by the panel will allow inputs from any number of domain experts and will also establish the methods of validating system responses.

Traceability of Requirements

Tracing requirements after they have been coded in rules may be more difficult than for conventional code, particularly when hybrid representation techniques are used, i.e. when both rules and objects are used to satisfy the program's requirements. This is an area that needs further consideration.

Verifying the Boundaries of the Expert System Domain

V&V of an expert system must be carefully aimed at identifying the boundaries of a problem since the experts sometimes can not readily do so. V&V must also ensure that the expert system fails gracefully in these circumstances.

There are additional issues not discussed in reference [4]. These are discussed more fully below.

Reasoning under Uncertainty

Some expert system applications deal with incomplete, inconsistent, or uncertain information. Humans do a very good job of reasoning under uncertainty, but it can be very difficult to develop consistent models which exactly duplicate this process. Numerous methods have been developed to allow expert systems to deal with this type of information, such as fuzzy logic, probability methods like Bayes theorem, Dempster-Schafer theory, certainty factors, etc. The nature of how humans use this type of information makes it very difficult to verify in an expert system. Different people

may give different answers when presented with the exact same information. V&V efforts must focus on two things; (1) verifying that the answers suggested in uncertain situations are 'acceptable' answers. The definition of 'acceptable' may be problem dependent, and (2) if uncertain information is combined, the method used to provide a certainty factor to the result must be consistent.

Maintaining a verifiable system

Long-term maintenance of an expert system is a poorly understood topic, primarily because there is little actual experience in this area. Soloway, et al. [5] discuss some of the difficulties in maintaining XCON, one of the largest and oldest expert systems in use today. They point out that XCON is a very dynamic system, with extensive changes occurring regularly. As with conventional software, most expert systems will change and V&V must be performed each time the modified system is released. The nature of almost all rule-based languages makes true modularization of code more difficult than with conventional software. Therefore, rule-based systems presently require complete retesting with every release, using a library of test cases. Good programming practices such as using explicit control features and simple rules are important aids, but may not be sufficient to prevent extensive retesting. This area will be better understood when more applications reach maintenance stages.

CONCLUSIONS

Verification and validation of expert systems is very important for the future success of this technology. Software will never be used in non-trivial applications unless the program developers can assure both users and managers that the software is reliable and generally free from error. Therefore, V&V of expert systems must be done. Although there are issues inherent to expert systems which introduce new complexities to the process, verification and validation can be done. The primary hindrance to effective V&V is the use of methodologies which do not produce testable requirements. Without requirements, V&V are meaningless concepts. An extension of the flight technique panels used in previous NASA programs should provide both documented requirements and very high levels of verification for expert systems.

REFERENCES

[1] Bochsler, D.C. and Goodwin, M.A., "Software Engineering Techniques Used to Develop an Expert System for Automated Space Vehicle Rendezvous", Proceeding of the Second Annual Workshop on Robotics and Expert Systems, Instrument Society of America, Research Triangle Park, NC., June 1986,

[2] Culbert, C.J., Riley, G., and Savely, R.T., "An Expert System Development Methodology Which Supports Verification and Validation", to be published.

[3] Stachowitz, R.A. and Combs, J.B., "Validation of Expert Systems", Proceedings Hawaii International Conference on Systems Sciences, Kona, Hawaii, January 6-9, 1987.

[4] Culbert, C.J., Riley, G., and Savely, R.T., "Approaches to the Verification of Rule-based Expert Systems", Proceedings of SOAR'87: Space Operations-Automation and Robotics Conference, Houston, TX., August 1987.

[5] Soloway, E., Bachant, J., and Jensen, K., "Assessing the Maintainability of XCON-in_RIME: Coping with the Problems of a VERY large Rule-Base", Proceedings of AAAI-87, Sixth National Conference on Artificial Intelligence, Seattle, WA., July 1987.